.

1.0 Introduction

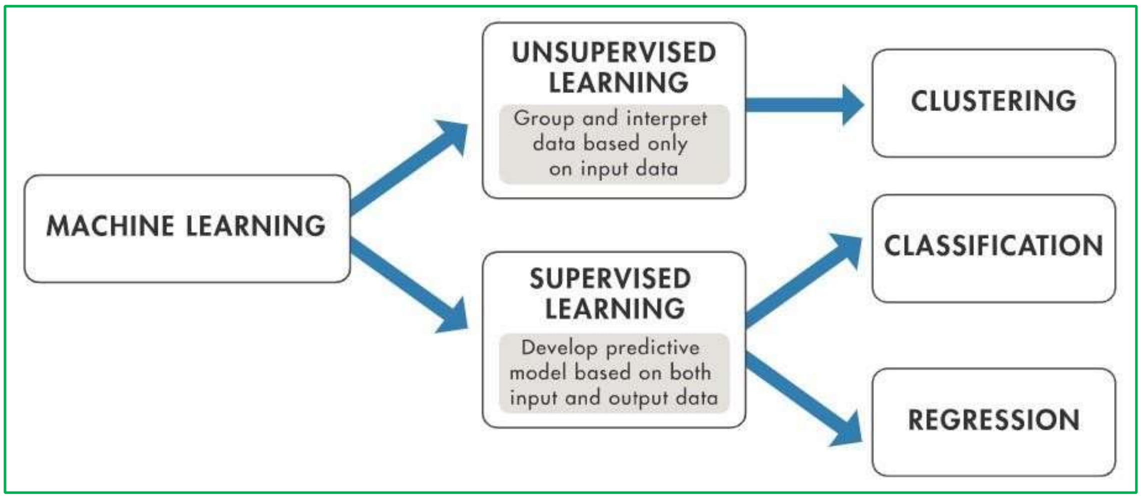
With the increasing power of computer technology, companies and institutions can nowadays store large amounts of data at a reduced cost. The amount of available data is increasing exponentially, and cheap disk storage makes it easy to store data that previously was thrown away. There is a huge amount of information locked up in databases that are potentially important but has not yet been explored. The growing size and complexity of the databases make it hard to analyze the data manually, so it is important to have automated systems to support the process. Hence there is a need for computational tools able to treat these large amounts of data and extract valuable information.

In this context, Data Mining provides automated systems capable of processing large amounts of data that are already present in databases. Data Mining is used to automatically extract important patterns and trends from databases seeking regularities or patterns that can reveal the structure of the data and answer business problems. Data Mining includes learning techniques that fall into the field of Machine learning. The growth of databases in recent years brings data mining to the forefront of new business technologies. Akin to the use of big data in various domains, the technology has a significant role in financial services and can be used to accurately and timely predict the possibilities of risks.

The goal of this program is to see how well various statistical methods perform in classifying whether a customer will subscribe to a term deposit or not given customer relationship data detail several of factors will determine y among them a job,a loan, campaign, etc are present. The data set has information about customers and campaigns done by the bank towards term deposits. There are column names y in the data set which tell whether the customer opted term deposit.

1.1. What are the different types of Machine Learning?

Machine learning uses two types of techniques: **Supervised learning**, which trains a model on known input and output data so that it can predict future outputs, and **Unsupervised learning**, which finds hidden patterns or intrinsic structures in input data.


MACHINE LEARNING
UNSUPERVISED LEARNING
Group and interpret data based only on input data
SUPERVISED LEARNING
Develop predictive model based on both input and output data
CLUSTERING
CLASSIFICATION
REGRESSION

Supervised Learning

Supervised machine learning builds a model that makes predictions based on evidence in the presence of uncertainty. A supervised learning algorithm takes a known set of input data and known responses to the data (output) and trains a model to generate reasonable predictions for the response to new data.

For example, in the healthcare industry supervised learning is used to predict heart attacks most probably.

Supervised learning uses **Regression** and **Classification** techniques to develop predictive models.

Regression techniques predict continuous responses. Common regression algorithms include linear model, nonlinear model, stepwise regression, Gradient Descent Regression, Support Vector Regression, and Ridge and Lasso Regressions.

Classification techniques predict discrete responses. Classification models classify input data. Common algorithms for performing classification include support vector machine (SVM), boosted and bagged decision trees, k-nearest neighbour, Naïve Bayes, discriminant analysis, logistic regression, and neural networks.

Unsupervised Learning

Unsupervised learning finds hidden patterns or intrinsic structures in data. It is used to draw inferences from datasets consisting of input data without labelled responses.

Clustering is the most common unsupervised learning technique. It is used for exploratory data analysis to find hidden patterns or groupings in data. Common algorithms for performing clustering include k-means and k-medoids, Apriori algorithms, hierarchical clustering, Gaussian mixture models, and hidden Markov models.

1.2. Benefits of Using Machine Learning

- **Automation of Everything:** Machine Learning is responsible for cutting the workload and time. By automating things, we let the algorithm do the hard work for us. Automation is now being done almost everywhere. The reason is that it is very reliable. Also, it helps us to think more creatively. Due to ML, we are now designing more advanced computers. These computers can handle various Machine Learning models and algorithms efficiently. Even though automation is spreading fast, we still don't completely rely on it. ML is slowly transforming the industry with its automation.

- **Wide Range of Applications:** ML has a wide variety of applications. This means that we can apply ML on any of the major fields. ML has its role everywhere from medical, business, banking to science and tech. This helps to create more opportunities. It plays a major role in customer interactions. Machine Learning can help in the detection of diseases more quickly. It is helping to lift up businesses. That is why investing in ML technology is worth it.

- **Efficient Handling of Data:** Machine Learning has many factors that make it reliable. One of them is data handling. ML plays the biggest role when it comes to data currently. It can handle any type of data. Machine Learning can be multidimensional or different types of data. It can process and analyse these data those normal systems can't. Data is the most important part of any Machine Learning model. Also, studying and handling of data is a field.
- **Scope of Improvement:** Machine Learning is the type of technology that keeps on evolving.

There is a lot of scope in ML to become the top technology in the future. The reason is it has a lot of research areas in it. This helps us to improve both hardware and software. In hardware, we have various laptops and GPU. These have various ML and Deep

Learning networks in them. These help in the faster processing power of the system. When it comes to software, we have various UI and libraries in use. These help in designing more efficient algorithms.

1.3. About Industry(Airlines Delays)

Airline delays are a common issue in the aviation industry and can result from various factors.These delays can be inconvenient for passengers , costly for airlines, and have a cascading effect on the entire system. Here are some key aspects to consider regarding airline delays:

- Passenger and Mitigation Impact
- Regulations and Compensation
- Mitigation Strategies
- Global Impact

1.3.1. AI / ML Role in Airlines Delays

Artificial Intelligence and Machine Learning (AIML) can play several significant roles in airlines delay prediction and management:

- Data Analysis: AIML algorithms can analyze historical flight data, weather patterns, air traffic, and other relevant information. This analysis helps in identifying trends and factors contributing to delays.
- Predictive Models: Machine Learning models can be trained to predict flight delays. These models consider various input variables, such as weather conditions ,airport congestion, historical data, and even real time factors like crew availability. Predictive models can provide estimates of potentials delays , allowing airlines to take proactive measures.
- Optimized scheduling: Airlines can use ML algorithms to optimize flight schedules ,taking into account factors that affect delays. For example historical data shows that certainroots are obtained delayed due to specific reasons ,airlines can adjust their schedules to mitigate their issues
- Crew and Resourse management: AI systems can help optimized crew assesments and other resources to ensure that they are in the right place at the right time,reducing delays caused by crew unavailability or last minute changes.

2.1 Main Drivers for AI Airlines

The main drivers for the adoption of artificial intelligence (AI) in the airline industry are diverse and cover various aspects of airline operations and customer service. Some of the key drivers include:

- Operational Efficiency: AI can help airlines optimize various operational aspects, such as flight scheduling, crew management, and maintenance. Predictive analytics and AI-driven solutions can reduce delays and improve overall operational efficiency.
- Cost Reduction: Airlines can use AI for cost reduction by automating routine tasks, predictive maintenance to reduce maintenance costs, and optimizing fuel consumption to save on operational expenses.
- Customer Experience: AI can enhance the customer experience through personalized recommendations, chatbots for customer service, and predictive analytics for better understanding customer preferences. This can lead to increased customer satisfaction and loyalty.
- Revenue Generation: AI can be used to optimize pricing strategies, demand forecasting, and dynamic pricing, which can maximize revenue for airlines. Additionally, AI-driven marketing campaigns can target the right audience and increase sales.
- Safety and Maintenance: AI can help improve safety by analyzing data from various sources, including sensors on aircraft, to detect potential maintenance issues before they become critical. This predictive maintenance can reduce the risk of accidents and increase overall safety.
- Predictive Analytics: AI can process vast amounts of data to make predictions about future events, such as predicting delays due to weather or other factors, allowing airlines to proactively manage these issues.
- Crew Management: AI can assist in optimizing crew schedules, ensuring that the right personnel are available when needed, and helping manage crew fatigue to improve both safety and efficiency.
- Baggage Handling: AI can help streamline baggage handling operations, reducing the likelihood of lost luggage and improving overall efficiency.

2.2 Internship Project - Data Link

The internship project data has been taken from OpenML and the link is

https://www.kaggle.com/datasets/jimschacko/airlines-dataset-to-predict-a-delay

Number of rows:539383

Number of columns: 9

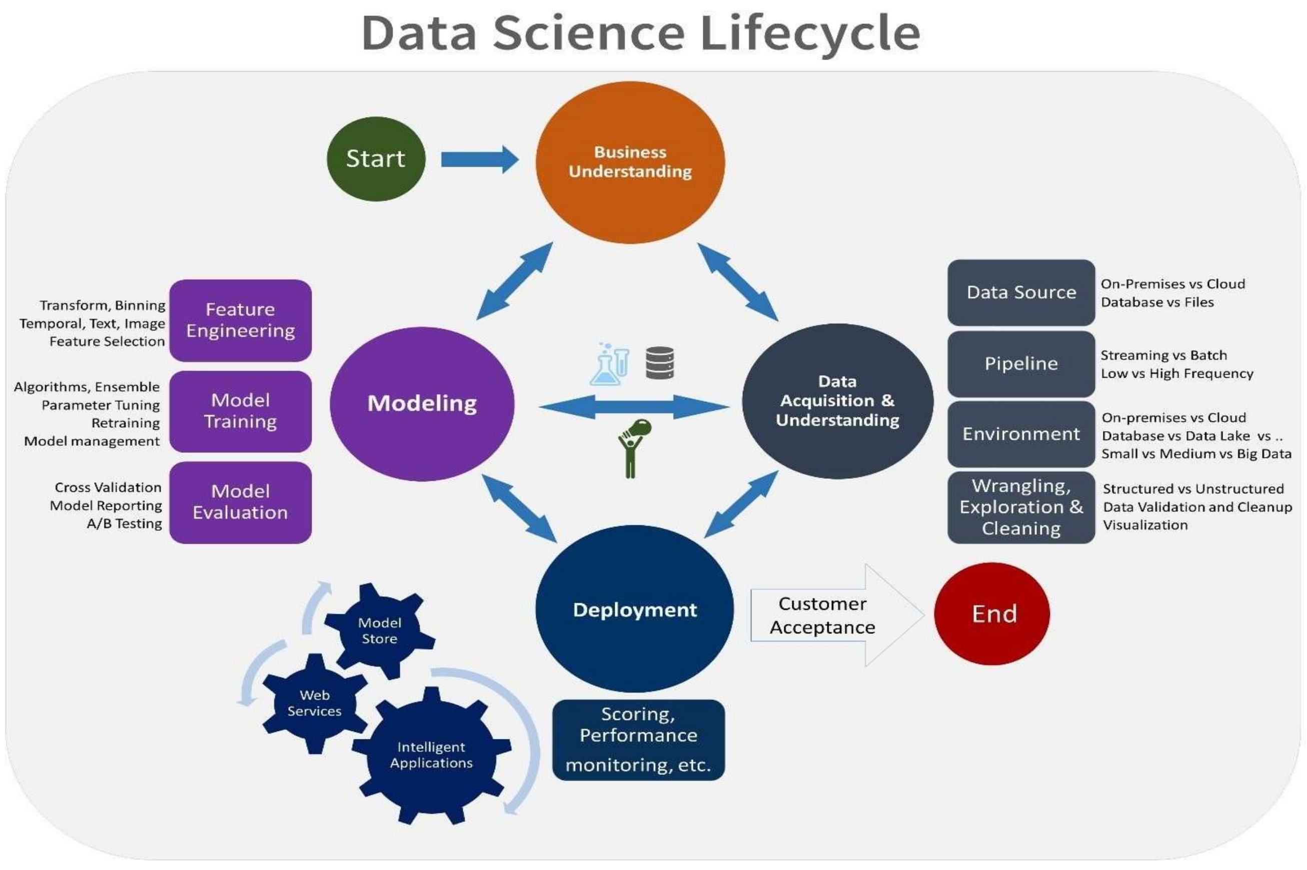
3.0. AI / ML Modelling and Results

3.1. Your Problem Statement

The problem statement here is to predict the delays in airlines and their management. Apply a suitable machine learning algorithm and predict the delays in the airlines (Flight delays). We are given the data of factors causing delays in airlines. The classification goal is to predict which flights are running with delays. This case study is inspired by this research paper where the researchers have used a very similar dataset as the one we will be using throughout this case study for predicting the delays in Airlines.

3.2. Data Science Project Life Cycle

Data Science is a multidisciplinary field of study that combines programming skills, domain expertise, and knowledge of statistics and mathematics to extract useful insights and knowledge from data.



3.2.1. Data Exploratory Analysis

Exploratory data analysis has been done on the data to look for relationship and correlation between different variables and to understand how they impact on target variable.

The explanatory analysis is done for the Airlines delay dataset– for a term Delay.

3.2.2. Data Pre-processing

We removed variables that do not affect our target variable (y) as they may add noise and also increase our computation time. We checked the data for anomalous data points and outliers. We did the principal component analysis on the data set to filter out unnecessary variables and to select only the important variables which have a greater correlation with our target variable.

3.2.2.1. Check the Duplicate and low variation data

To check Low Variation Data:

The variance can be used as a filter for identifying columns to be removed from the dataset. A column that has a single value has a variance of 0.0, and a column that has very few unique values will have a small variance value. The Variance Threshold class from the tsci-kitkit-learn library supports this as a type of feature selection. An instance of the class can be created to specify the "*threshold*" argument, which defaults to 0.0 to remove columns with a single value. It can then be fitted and applied to a dataset by calling the *fit_transform()* function to create a transformed version of the dataset where the columns that have a variance lower than the threshold have been removed automatically.

To check Duplicate Data:

You have a dataset and have to check whether there are duplicates or not. The Python pandas library has a method for it, that is duplicated(). It checks for duplicate rows and returns True and False.

3.2.2.2. Identify and address the missing variables

Analyze each column with missing values carefully to understand the reasons behind missing values as it is crucial to find out a strategy for handling the missing values.

data.isnull().values.any()

It prints true if any missing value is available and false if no missing data.

We handle these missing variables using 2 techniques

1. case deletion

2. data imputation **Case deletion:**

Listwise deletion means that any individual in a data set is deleted from analysis if they're missing data on *any* variable in the analysis.

Listwise deletion works when

- Data is missing completely at random (MCAR)
- You have sufficient power anyway, even though you lost part of your data set

Listwise deletion works when

- The data are missing completely at random (MCAR)
- You have sufficient power anyway, even though you lost part of your data set

The below code is used to delete all rows that have null values. data.dropna()

The below code is used for dropping columns that have missing values data.dropna(axis=1) The below code is used to fill all the missing values by 0 data.fillna()

Data Imputation:

Imputation is the process of replacing missing **data** with substituted values. When substituting for a **data** point.

Few imputation methods:

Mean/Median/Mode Substitution:

The choice for missing values that are known not to be zero is to use some *central values such as* mean, median, or mode. We might choose to use the mean if the variable is generally normally distributed. If the data exhibit skewness then the median might be a better choice. For categoric variables, we may choose to use the mode (the most frequent value) as the default to fill in for the missing values and numeric variables. It is a blunt approach to imputation and can lead to poor performance.

data.fillna(data.mean()) data.fillna(data.median()) data.fillna(data.mode())

The methods mentioned below will be better.

Substitution:

Fill the value from a different subject that is outside of the data.

Hot deck imputation: A randomly chosen value from an individual in the sample who has similar values on other variables is filled.

Regression imputation: The predicted value obtained by regressing the missing variable on other variables is filled.

3.2.2.3. Handling of Outliers

Deleting the values: You can delete the outliers if you know that the outliers are wrong or if the reason the outlier was created is never going to happen in the future. For example, there is a data set of people's ages and the usual ages lie between 0 to 90 but there is data entry off the age 150 which is nearly impossible. So, we can safely drop the value that is 150.

Changing the values: We can also change the values in the cases when we know the reason for the outliers. Consider the previous example for measurement or instrument errors where we had 10 voltmeters out of which one voltmeter was faulty. Here what we can do is we can take another set of readings using a correct voltmeter and replace them with the readings that were taken by the faulty voltmeter.

Data transformation: Data transformation is useful when we are dealing with highly skewed data sets. By transforming the variables, we can eliminate the outliers for example taking the natural log of a value reduces the variation caused by the extreme values. This can also be done for data sets that do not have negative values.

Using different analysis methods: You could also use different statistical tests that are not as much impacted by the presence of outliers.

Valuing the outliers: In case there is a valid reason for the outlier to exist and it is a part of our natural process, we should investigate the cause of the outlier as it can provide valuable clues that can help you better understand your process performance. Outliers may be hiding precious information that could be invaluable to improving your process performance. You need to take the time to understand the special causes that contributed to these outliers. Fixing these special causes can give you a significant boost in your process performance and improve customer satisfaction.

3.2.2.4. Categorical data and Encoding Techniques

Categorical variables are usually represented as ‘strings’ or ‘categories’ and are finite in number. Here are a few examples:

1. The city where a person lives: Delhi, Mumbai, Ahmedabad, Bangalore, etc.
2. The department a person works in: Finance, Human resources, IT, Production.

The variables only have definite possible values. Further, we can see there are two kinds of categorical data-

- **Ordinal Data:** The categories have an inherent order
- **Nominal Data:** The categories do not have an inherent order

In Ordinal data, while encoding, one should retain the information regarding the order in which the category is provided. Like in the above example the highest degree a person possesses, gives vital information about his qualification. The degree is an important feature to decide whether a person is suitable for a post or not.

While encoding Nominal data, we have to consider the presence or absence of a feature. In such a case, no notion of order is present. For example, the city a person lives in. For the data, it is important to retain where a person lives. Here, We do not have any order or sequence. It is equal if a person lives in Delhi or Bangalore.

Label Encoding or Ordinal Encoding

We use this categorical data encoding technique when the categorical feature is ordinal. In this case, retaining the order is important. Hence encoding should reflect the sequence.

In Label encoding, each label is converted into an integer value. We will create a variable that contains the categories representing the education qualification of a person.

Binary Encoding

Binary encoding is a combination of Hash encoding and one-hot encoding. In this encoding scheme, the categorical feature is first converted into numerical using an ordinal encoder. Then the numbers are transformed into binary numbers. After that binary value is split into different columns.

Binary encoding works well when there are a high number of categories. For example the cities in a country where a company supplies its products.

Binary encoding is a memory-efficient encoding scheme as it uses fewer features than one-hot encoding. Further, It reduces the curse of dimensionality for data with high cardinality.

3.2.2.5. Feature Scaling

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing. More specifically, we will be looking at 3 different scalers in the Scikit-learn library for feature scaling and they are:

1.MinmaxScalar **2.** Standard Scalar **3.** Robust Scalar

The normalization, also known as min-max scaling, is a scaling technique whereby the values in a column are shifted so that they are bounded between a fixed range of 0 and 1. Minmax Scalar is the Scikit-learn function for normalization. On the other hand, standardization or Z-score normalization is another scaling technique whereby the values in a column are rescaled so that they demonstrate the properties of standard Gaussian distribution, that is mean = 0 and variance = 1. Standard Scalar is the Scikit-learn function for standardization. Unlike StandardScaler, Robust Scalar scales feature using statistics that are robust to outliers. More specifically, Robust Scaler removes the median and scales the data according to the interquartile range, thus making it less susceptible to outliers in the data.

3.2.3. Selection of Dependent and Independent variables

The dependent or target variable here is Delay or not the target variable is selected based on our problem and what we are trying to predict.

The independent variables are selected after doing exploratory data analysis and we used Boruta to select which variables are most affecting our target variable.

3.2.4. Data Sampling Methods

The data we have is highly unbalanced so we used some sampling methods which are used to balance the target variable so that our model will be developed with good accuracy and precision. We used three Sampling methods

3.2.4.1. Stratified sampling

Stratified sampling randomly selects data points from the majority class so they will be equal to the data points in the minority class. So, after the sampling, both the class will have the same no of observations.

It can be performed using the strata function from the library sampling.

3.2.4.2. Simple random sampling

Simple random sampling is a sampling technique where a set percentage of the data is selected randomly. It is generally done to reduce bias in the dataset which can occur if data is selected manually without randomizing the dataset.

We used this method to split the dataset into a training dataset that contains 70% of the total data and a test dataset with the remaining 30% of the data.

3.2.5. Models Used for Development

We built our predictive models by using the following ten algorithms

3.2.5.1. Model 01: Logistic

Logistic uses the logit link function to convert the likelihood values to probabilities so we can get a good estimate of the probability of a particular observation being a positive class or negative class. The also gives us the p-value of the variables which tells us about the significance of each independent variable.

3.2.5.2. Model 02: Random Forest

A random forest is an algorithm that consists of many decision trees. It was first developed by Leo Breiman and Adele Cutler. The idea behind it is to build several trees, to have the instance classified by each tree, and to give a "vote" at each class. The model uses a "bagging" approach and the random selection of features to build a collection of decision trees with controlled variance. The instance's class is the class with the highest number of votes, the class that occurs the most within the leaf in which the instance is placed.

The error of the forest depends on:

- Trees correlation: the higher the correlation, the higher the forest error rate. •

  The strength of each tree in the forest. A strong tree is a tree with low error. By using trees that classify the instances with low error the error rate of the forest decreases.

3.2.5.3. Model 03: ANN

Artificial neural networks can theoretically solve any problem. ANNs can identify hidden patterns between the variables and can find how different combinations of variables can affect the target variable. The error correction is done by gradient descent algorithm which can reduce the error rate as much as possible for the given data.

3.2.5.4. Model 04: Extra-trees classifier

This class implements a meta estimator that fits several randomized decision trees on various subsamples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

The Extra Trees algorithm works by creating a large number of unpruned decision trees from the training dataset. Predictions are made by averaging the prediction of the decision trees in the case of regression or using majority voting in the case of classification.

3.2.5.5. Model 05: Support Vector Machines (SVM)

Support Vector Machines are perhaps one of the most popular and talked about machine learning algorithms. A hyperplane is a line that splits the input variable space. In SVM, a hyperplane is selected to best separate the points in the input variable space by their class, either class 0 or class 1. In two dimensions, you can visualize this as a line, and let’s assume that all of our input points can be completely separated by this line. The SVM learning algorithm finds the coefficients that result in the best separation of the classes by the hyperplane.

3.2.5.6. Model 6: Light Gradient Boosting Machine (LightGBM)

LightGBM is a gradient boosting framework based on decision trees to increase the efficiency of the model and reduces memory usage. LightGBM splits the tree leaf-wise as opposed to other boosting algorithms that grow tree level-wise. It chooses the leaf with maximum delta loss to grow. Since the leaf is fixed, the leaf-wise algorithm has a lower loss compared to the level-wise algorithm. Leaf-wise tree growth might increase the complexity of the model and may lead to overfitting in small datasets.

3.2.5.7. Model 07: K-Means

A highly popular, high-speed algorithm, K-means involves placing unlabeled data points in separate groups based on similarities. This algorithm is used for the clustering model. Kmeans tries to figure out what the common characteristics are of individuals and groups them together. This is particularly helpful when you have a large data set and are looking to implement a personalized plan—this is very difficult to do with one million people.

In the context of predictive analytics for healthcare, a sample size of patients might be placed into five separate clusters by the algorithm. One particular group shares multiple characteristics: they don't exercise, they have an increasing hospital attendance record (three times one year and then ten times the next year), and they are all at risk for diabetes.

3.2.5.8. Model 08: K-nearest neighbours

The k-nearest neighbours algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

KNN works by finding the distances between a query and all the examples in the data, selecting the specified number of examples (K) closest to the query, then voting for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

3.2.5.9. Model 09: XGBoost

The GPU-accelerated XGBoost algorithm **makes use of fast parallel prefix sum operations to scan through all possible splits, as well as parallel radix sorting to repartition data**. It builds a decision tree for a given boosting iteration, one level at a time, processing the entire dataset concurrently on the GPU.

XGBoost minimizes a regularized (L1 and L2) objective function that combines a convex loss function (based on the difference between the predicted and target outputs) and a penalty term for model complexity (in other words, the regression tree functions).

3.2.5.10. Model 10: Naive Bayes

It works on the principle of the Bayes Theorem, which finds the probability of an event considering some true conditions. Bayes Theorem is represented as:

The algorithm is called Naive because it believes all variables are independent, and the presence of one variable doesn't have any relation to the other variables, which is never the case in real life. As a result, naive Bayes could be used in Email Spam classification and text classification.

3.3. AI / ML Models Analysis and Final Results

We used our train dataset to build the above models and used our test data to check the accuracy and performance of our models.

We used a confusion matrix to check the accuracy, Precision, Recall, and F1 score of our models and compare and select the best model for a given auto dataset of size ~ 272252 policies.

3.3.1. Different Model codes

The Python code for models with a simple random sampling technique is as follows:

3.3.2. KNN Algorithm

The Python code for models with a KNN Algorithm is as follows:

```
# Build KNN Model

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

import sklearn.metrics as metrics

from sklearn.metrics import roc_curve, roc_auc_score

accuracy = []

for a in range(1, 21, 1):

    k = a

 # Build the model

 ModelKNN = KNeighborsClassifier(n_neighbors=k)

 # Train the model

 ModelKNN.fit(x_train, y_train)

 # Predict the model

y_pred = ModelKNN.predict(x_test)
```

```
y_pred_prob = ModelKNN.predict_proba(x_test)

print('KNN_K_value = ', a)

# Print the model name

print('Model Name: ', ModelKNN)

# confusion matrix in sklearn

from sklearn.metrics import confusion_matrix

from sklearn.metrics import classification_report

 # actual values

actual = y_test

# predicted values

predicted = y_pred

# confusion matrix

matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)

print('Confusion matrix : \n', matrix)

 # outcome values order in sklearn

tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)

print('Outcome values : \n', tp, fn, fp, tn)

 # classification report for precision, recall f1-score and accuracy

C_Report = classification_report(actual,predicted,labels=[1,0])

print('Classification report : \n', C_Report)

# calculating the metrics

 sensitivity = round(tp/(tp+fn), 3);
```

```
specificity = round(tn/(tn+fp), 3);

accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);

balanced_accuracy = round((sensitivity+specificity)/2, 3);

 precision = round(tp/(tp+fp), 3);

f1Score = round((2*tp/(2*tp + fp + fn)), 3);

# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.

# A model with a score of +1 is a perfect model and -1 is a poor model

from math import sqrt

 #mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)

#MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)

MCC=0

 print('Accuracy :', round(accuracy*100, 2),'%')

 print('Precision :', round(precision*100, 2),'%')

 print('Recall :', round(sensitivity*100,2), '%')

 print('F1 Score :', f1Score)

 print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )

 print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')

 print('MCC :', MCC)

# Area under ROC curve

from sklearn.metrics import roc_curve, roc_auc_score

print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))

# ROC Curve
```

```
from sklearn.metrics import roc_auc_score

from sklearn.metrics import roc_curve

model_roc_auc = roc_auc_score(actual, predicted)

fpr, tpr, thresholds = roc_curve(actual, ModelKNN.predict_proba(x_test)[:,1])

plt.figure()

# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)

plt.plot(fpr, tpr, label= 'Classification Model' % model_roc_auc)

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver operating characteristic')

plt.legend(loc="lower right")

#plt.savefig('Log_ROC')

plt.show()

 #------------------------------------------------------------------------------

new_row = {'Model Name' : ModelKNN,

            'KNN K Value' : a,

            'True_Positive' : tp,

            'False_Negative' : fn,

            'False_Positive' : fp,
            'True_Negative' : tn,

            'Accuracy' : accuracy,

            'Precision' : precision,

            'Recall' : sensitivity,

            'F1 Score' : f1Score,

            'Specificity' : specificity,

            'MCC':MCC,

            'ROC_AUC_Score':roc_auc_score(actual, predicted),

            'Balanced Accuracy':balanced_accuracy}

       KNN_Results = KNN_Results.append(new_row, ignore_index=True)

       #------KNN_Results----------------------------------------------------------------------
```

3.3.3. Comparision of all Algorithms

```
# Build the Calssification models and compare the results

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import ExtraTreesClassifier

from sklearn.neighbors import KNeighborsClassifier

#from sklearn.svm import SVC

#from sklearn.naive_bayes import GaussianNB

# Create objects of classification algorithm with default hyper-

parameters

ModelLR = LogisticRegression()

ModelDC = DecisionTreeClassifier()
```

```
ModelRF = RandomForestClassifier()
ModelET = ExtraTreesClassifier()
ModelKNN = KNeighborsClassifier(n_neighbors=5)
#ModelSVM = SVC(kernel='rbf', random_state = 42,
class_weight='balanced', probability=True)
#ModelGNB = GaussianNB()
# Evalution matrix for all the algorithms
MM = [ModelLR, ModelDC, ModelRF, ModelET,
ModelKNN]
for models in MM:
# Fit the model
models.fit(x_train, y_train)
# Prediction
y_pred = models.predict(x_test)
y_pred_prob = models.predict_proba(x_test)
# Print the model name
print('Model Name: ', models)
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
```

```
matrix=confusion_matrix(actual,predicted,
labels=[1,0],sample_weight=None, normalize=None)
 print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp,fn,fp,tn=
confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
 print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and
accuracy
C_Report=
classification_report(actual,predicted,labels=[1,0])
 print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values
of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a
poor model
from math import sqrt
 #mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
#MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
```

```
    MCC=0
    print('Accuracy :', round(accuracy*100, 2),'%')
    print('Precision :', round(precision*100, 2),'%')
    print('Recall :', round(sensitivity*100,2), '%')
    print('F1 Score :', f1Score)
    print('Specificity or True Negative Rate :',
round(specificity*100,2), '%'  )
    print('Balanced Accuracy :',
round(balanced_accuracy*100, 2),'%')
    print('MCC :', MCC)
   # Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:',round(roc_auc_score(actual,
predicted), 3))
 # ROC Curve
 from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
Model_roc_auc = roc_auc_score(actual, predicted)
 fpr,tpr,thresholds = roc_curve(actual,
models.predict_proba(x_test)[:,1])
plt.figure()
 # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)'
% logit_roc_auc)
 plt.plot(fpr, tpr, label= 'Classification Model' %
Model_roc_auc)
 plt.plot([0, 1], [0, 1],'r--')
```

```
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")
    plt.savefig('Log_ROC')
    plt.show()
    print('--------------------------------------------------------------
----------------------------------------')
    #-----------------------------------------------------------------
----------------------------------------
    new_row = {'Model Name' : models,
               'True_Positive' : tp,
               'False_Negative' : fn,
               'False_Positive' : fp,
               'True_Negative' : tn,
               'Accuracy' : accuracy,
               'Precision' : precision,
               'Recall' : sensitivity,
               'F1 Score' : f1Score,
               'Specificity' : specificity,
               'MCC':MCC,
               'ROC_AUC_Score':roc_auc_score(actual,
predicted),
               'Balanced Accuracy':balanced_accuracy}
    EM_Results = EM_Results.append(new_row,
ignore_index=True)
    #------------------------------------------------------------------
-----------------------------------
```

Stratified Sampling: Random Forest model performance is good, by considering the confused matrix, highest accuracy (1.0) &good F1 score (1.0). This is because random forest uses bootstrap aggregation which can reduce bias and variance in the data and can lead to good predictions with the bank marketing dataset.

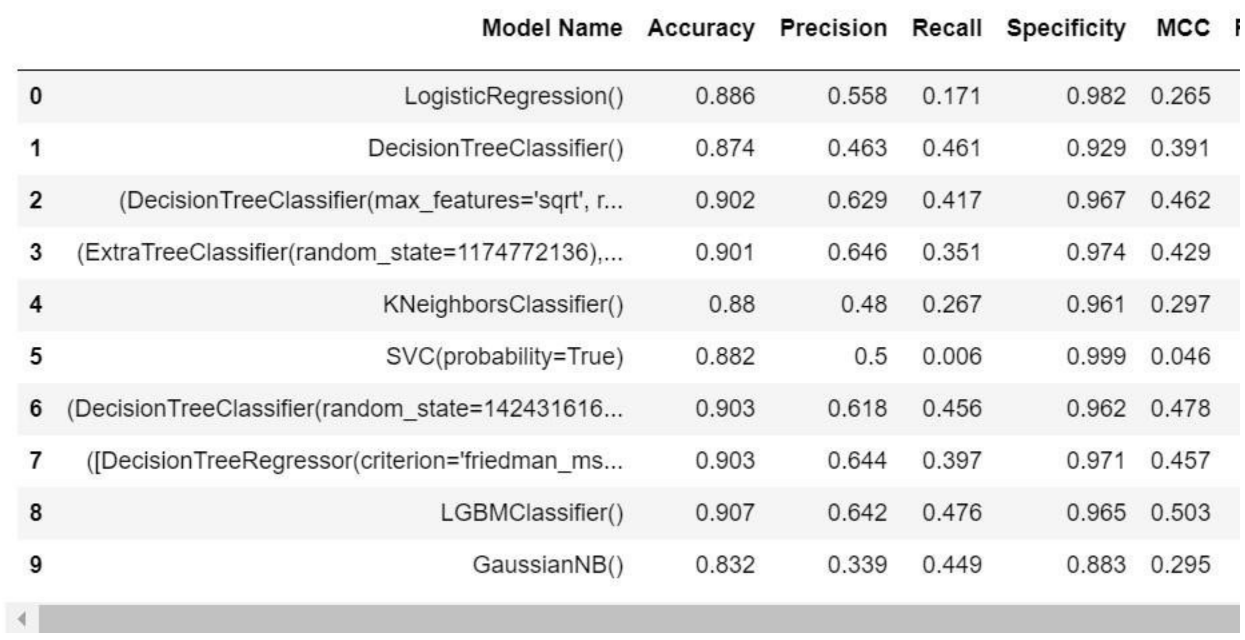
Simple Random Sampling: Artificial Neural Networks / Random Forest is out performed Logistic Regression model, by considering the confused matrix, highest accuracy (1.0) &good F1 score (1.0). This is because Artificial Neural Networks have hidden and complex patterns between different variables and can lead to good predictions with claims datasets.

4.0. Conclusions and Future work

The model results in the following order by considering the model accuracy, F1 score, and RoC AUC score.

1) **LightGBM** with Random Sampling
2) **Decision Tree** with Simple Random Sampling
3) **Random Forest** with Simple Random Sampling

We recommend model - **LightGBM** with the Random Sampling technique as the best fit for the given bank marketing dataset. We considered LightGBM because it can reduce bias and variance in the data and can lead to good predictions with the dataset.



	Model Name	Accuracy	Precision	Recall	Specificity	MCC
0	LogisticRegression()	0.886	0.558	0.171	0.982	0.265
1	DecisionTreeClassifier()	0.874	0.463	0.461	0.929	0.391
2	(DecisionTreeClassifier(max_features='sqrt', r...	0.902	0.629	0.417	0.967	0.462
3	(ExtraTreeClassifier(random_state=1174772136),...	0.901	0.646	0.351	0.974	0.429
4	KNeighborsClassifier()	0.88	0.48	0.267	0.961	0.297
5	SVC(probability=True)	0.882	0.5	0.006	0.999	0.046
6	(DecisionTreeClassifier(random_state=142431616...	0.903	0.618	0.456	0.962	0.478
7	([DecisionTreeRegressor(criterion='friedman_ms...	0.903	0.644	0.397	0.971	0.457
8	LGBMClassifier()	0.907	0.642	0.476	0.965	0.503
9	GaussianNB()	0.832	0.339	0.449	0.883	0.295

5.0. References

- https://www.kaggle.com/datasets/jimschacko/airlines-dataset-to-predict-a-delay

6.0. Appendices

6.1. Python Code Results

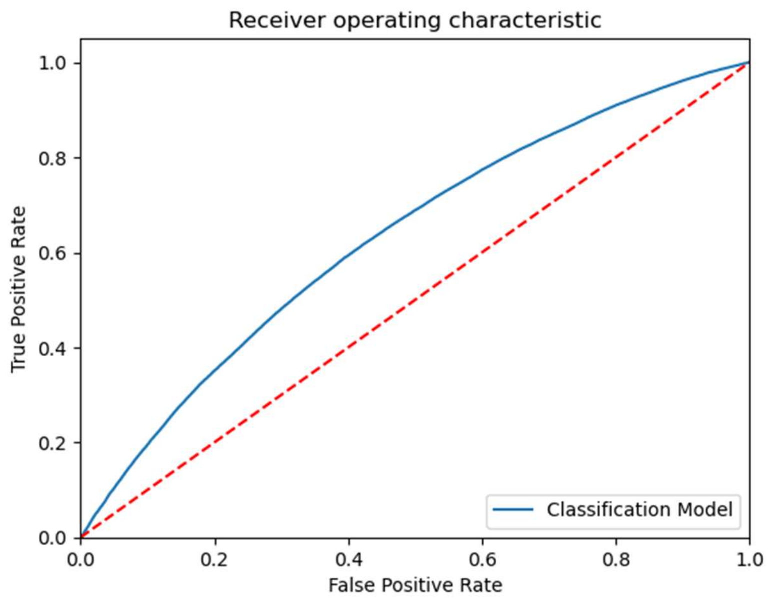
Logistic regression output:

```
Model Name:  LogisticRegression()
Confusion matrix :
 [[29545 42365]
 [22099 67806]]
Outcome values :
 29545 42365 22099 67806
Classification report :
               precision    recall  f1-score   support

           1       0.57      0.41      0.48     71910
           0       0.62      0.75      0.68     89905

    accuracy                           0.60    161815
   macro avg       0.59      0.58      0.58    161815
weighted avg       0.60      0.60      0.59    161815

Accuracy : 60.2 %
Precision : 57.2 %
Recall : 41.1 %
F1 Score : 0.478
Specificity or True Negative Rate : 75.4 %
Balanced Accuracy : 58.2 %
MCC : 0
roc_auc_score: 0.583
```



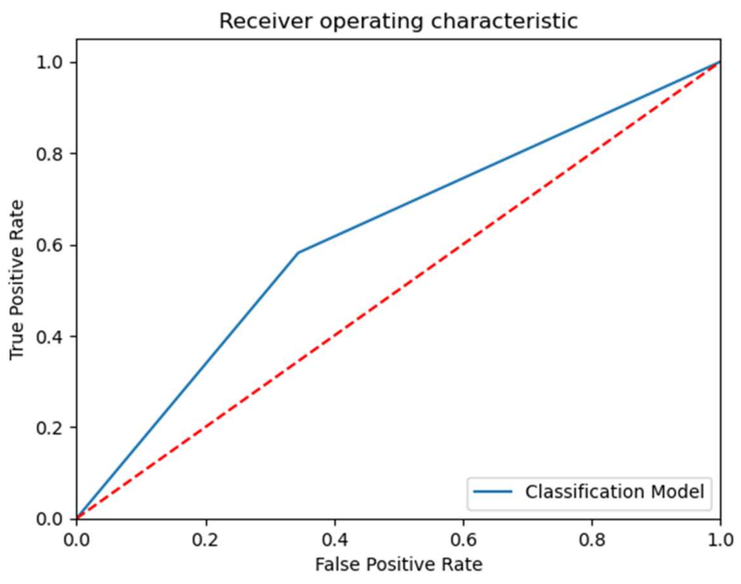
Decision tree classifier output:

```
 Model Name:  DecisionTreeClassifier()
Confusion matrix :
 [[41822 30088]
 [30959 58946]]
Outcome values :
 41822 30088 30959 58946
Classification report :
               precision    recall  f1-score   support

           1       0.57      0.58      0.58     71910
           0       0.66      0.66      0.66     89905

    accuracy                           0.62    161815
   macro avg       0.62      0.62      0.62    161815
weighted avg       0.62      0.62      0.62    161815

Accuracy : 62.3 %
Precision : 57.5 %
Recall : 58.2 %
F1 Score : 0.578
Specificity or True Negative Rate : 65.6 %
Balanced Accuracy : 61.9 %
MCC : 0
roc_auc_score: 0.619
```



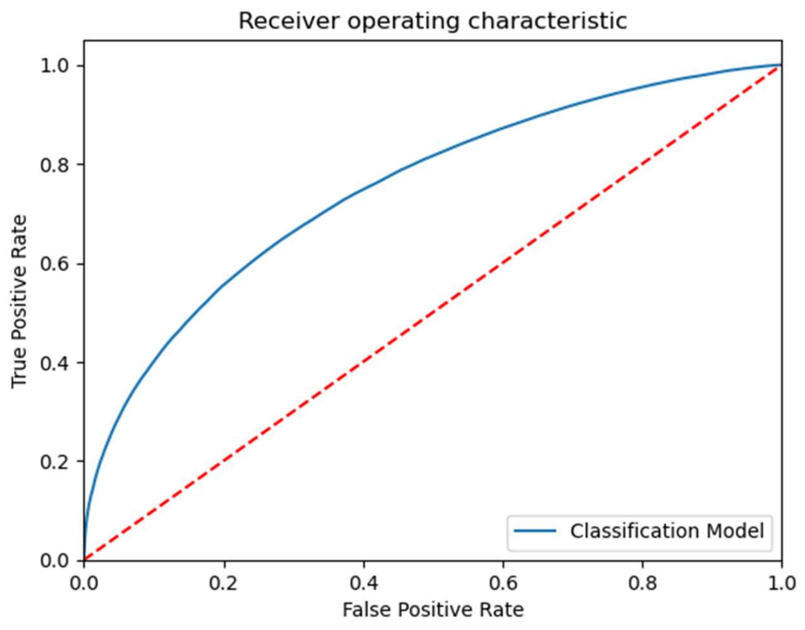
Random Forest Classifier Output:

```
Model Name:  RandomForestClassifier()
Confusion matrix :
 [[42630 29280]
 [20894 69011]]
Outcome values :
 42630 29280 20894 69011
Classification report :
               precision    recall  f1-score   support

           1       0.67      0.59      0.63     71910
           0       0.70      0.77      0.73     89905

    accuracy                           0.69    161815
   macro avg       0.69      0.68      0.68    161815
weighted avg       0.69      0.69      0.69    161815

Accuracy : 69.0 %
Precision : 67.1 %
Recall : 59.3 %
F1 Score : 0.63
Specificity or True Negative Rate : 76.8 %
Balanced Accuracy : 68.0 %
MCC : 0
roc_auc_score: 0.68
```



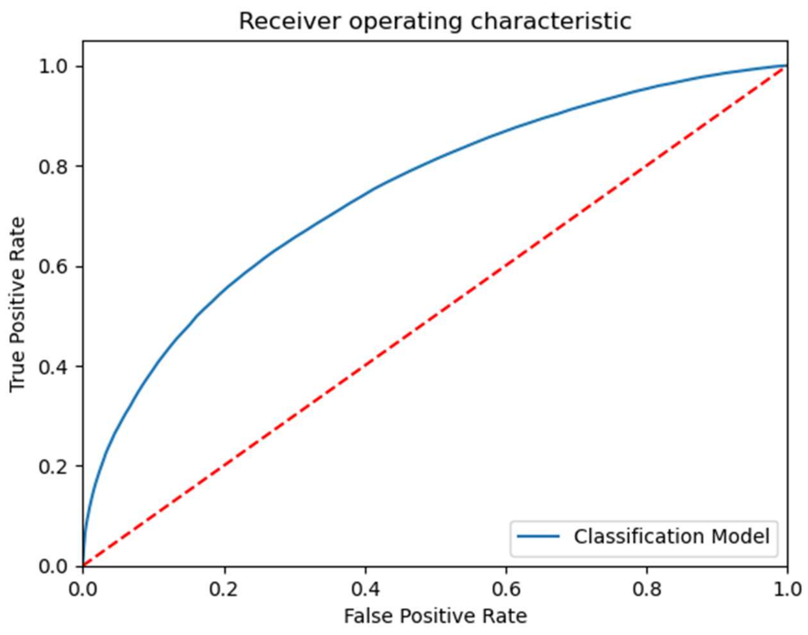
Extra Tree Classifier Output:

```
Model Name:  ExtraTreesClassifier()
Confusion matrix :
 [[42131 29779]
 [20735 69170]]
Outcome values :
 42131 29779 20735 69170
Classification report :
               precision    recall  f1-score   support

           1       0.67      0.59      0.63     71910
           0       0.70      0.77      0.73     89905

    accuracy                           0.69    161815
   macro avg       0.68      0.68      0.68    161815
weighted avg       0.69      0.69      0.68    161815

Accuracy : 68.8 %
Precision : 67.0 %
Recall : 58.6 %
F1 Score : 0.625
Specificity or True Negative Rate : 76.9 %
Balanced Accuracy : 67.8 %
MCC : 0
roc_auc_score: 0.678
```



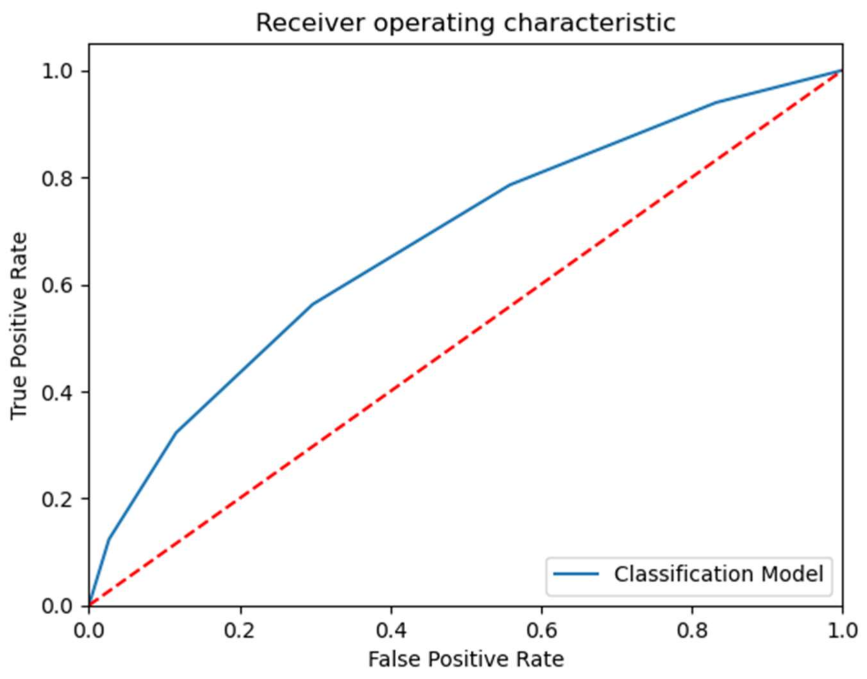
K Neighbour Classifier Output:

```
Model Name:  KNeighborsClassifier()
Confusion matrix :
 [[40460 31450]
 [26701 63204]]
Outcome values :
 40460 31450 26701 63204
Classification report :
               precision    recall  f1-score   support

           1       0.60      0.56      0.58     71910
           0       0.67      0.70      0.68     89905

    accuracy                           0.64    161815
   macro avg       0.64      0.63      0.63    161815
weighted avg       0.64      0.64      0.64    161815

Accuracy : 64.1 %
Precision : 60.2 %
Recall : 56.3 %
F1 Score : 0.582
Specificity or True Negative Rate : 70.3 %
Balanced Accuracy : 63.3 %
MCC : 0
roc_auc_score: 0.633
```


Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Classification Model